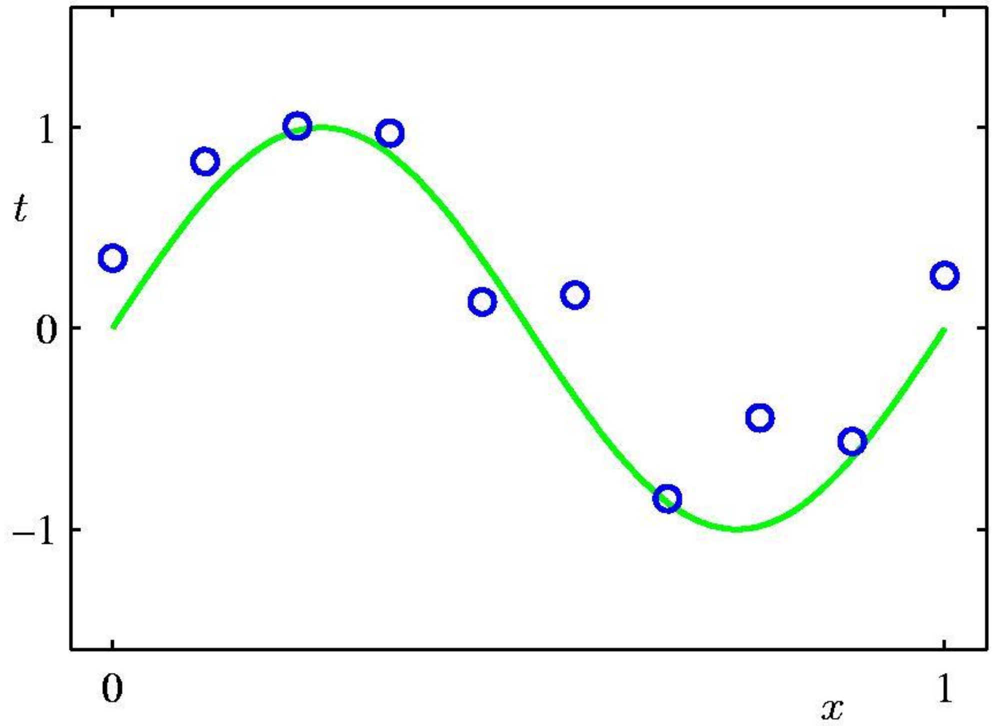
REGRESSION PART 3: REGULARIZATION

Hsin-Min Lu
盧信銘
台大資管系

Should We Always Prefer More Features?

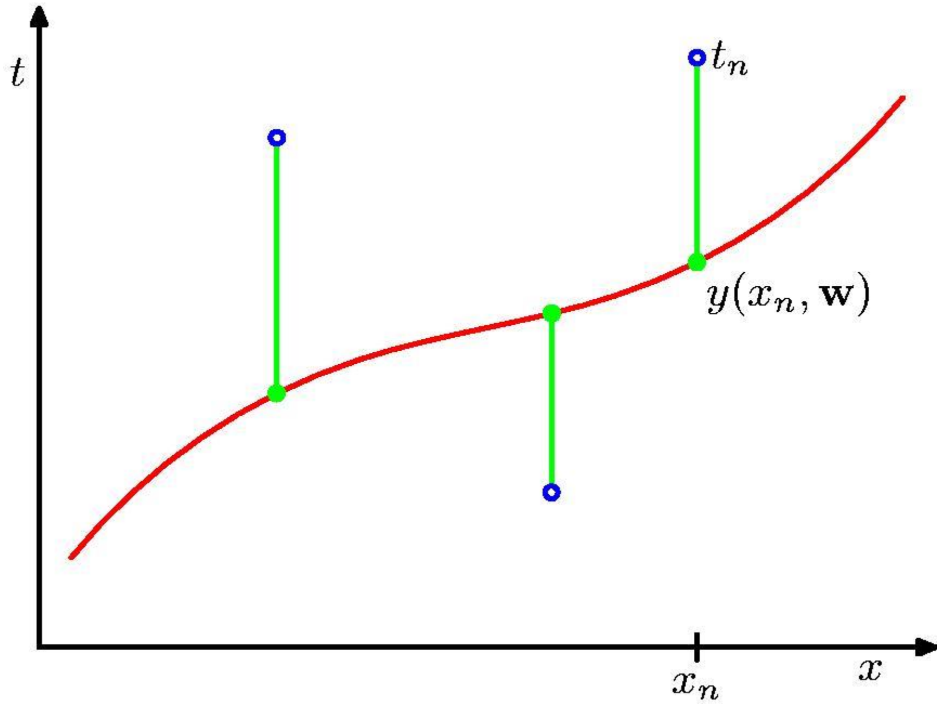
- In the previous example, we have seen that additional features allow us to capture additional variations of the outcome, and thus provides better prediction.
- The key question is should we always prefer more features when constructing a model?
- Is there any drawback when we add large amounts of features?

Polynomial Curve Fitting



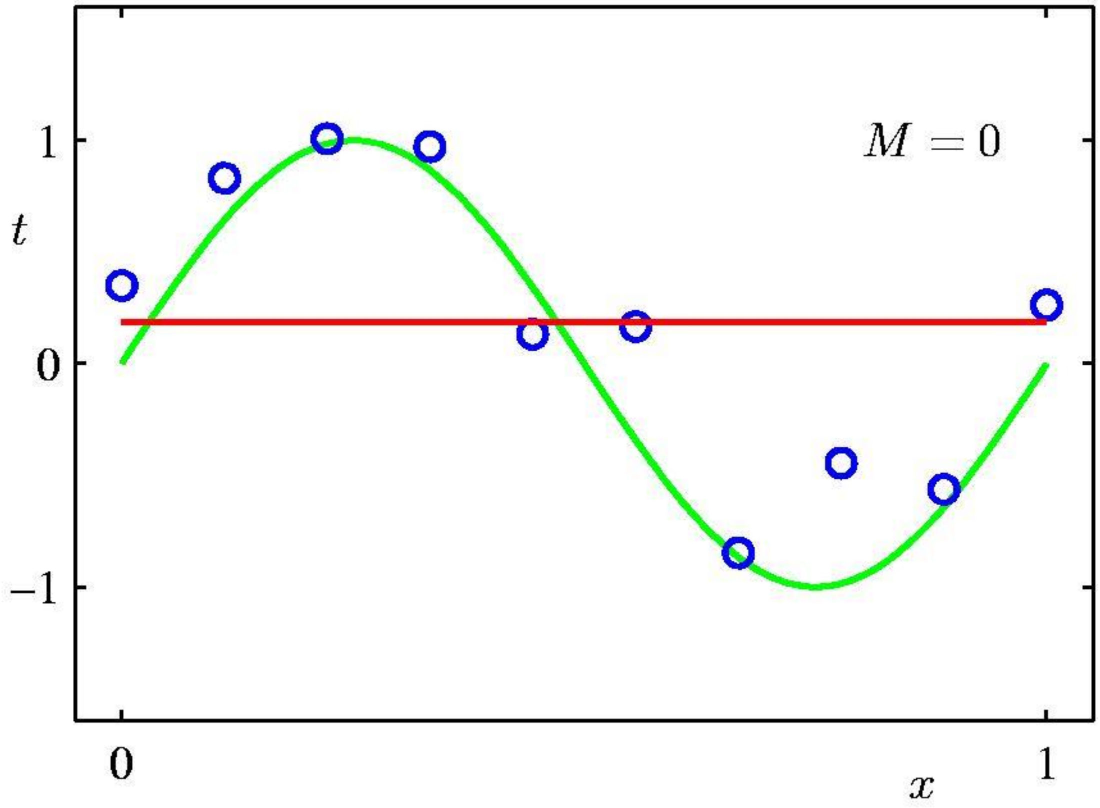
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Sum-of-Squares Error Function

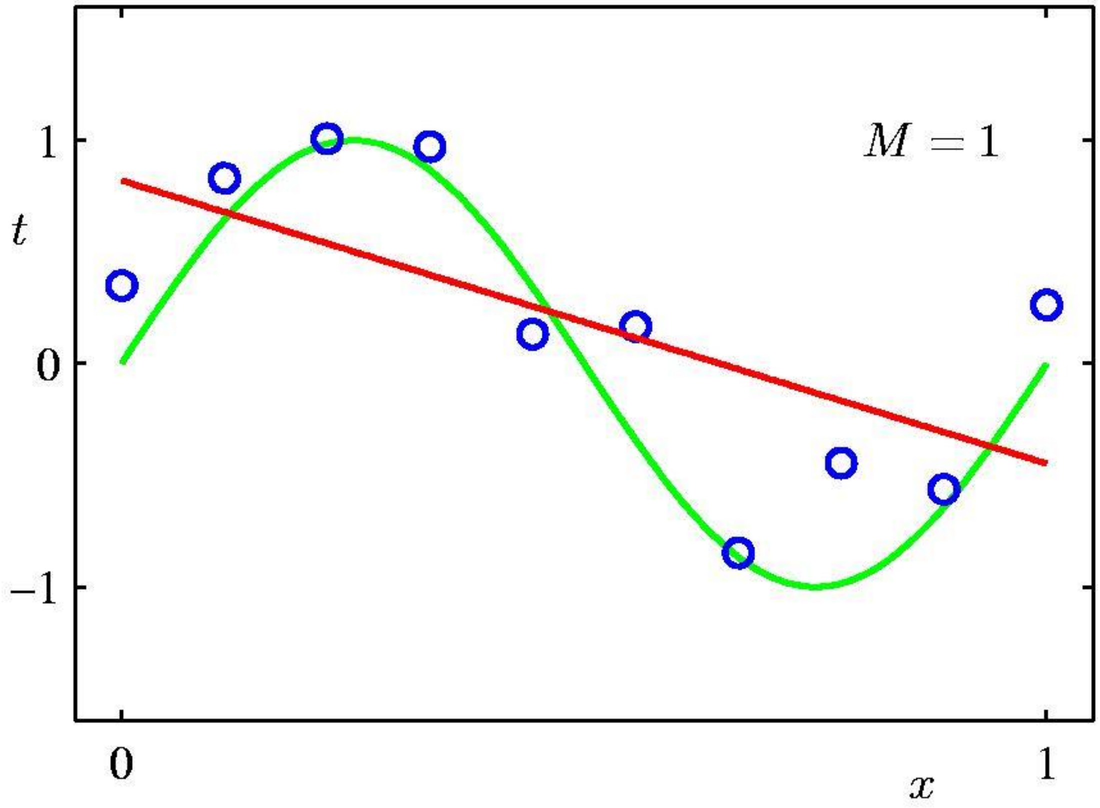


$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

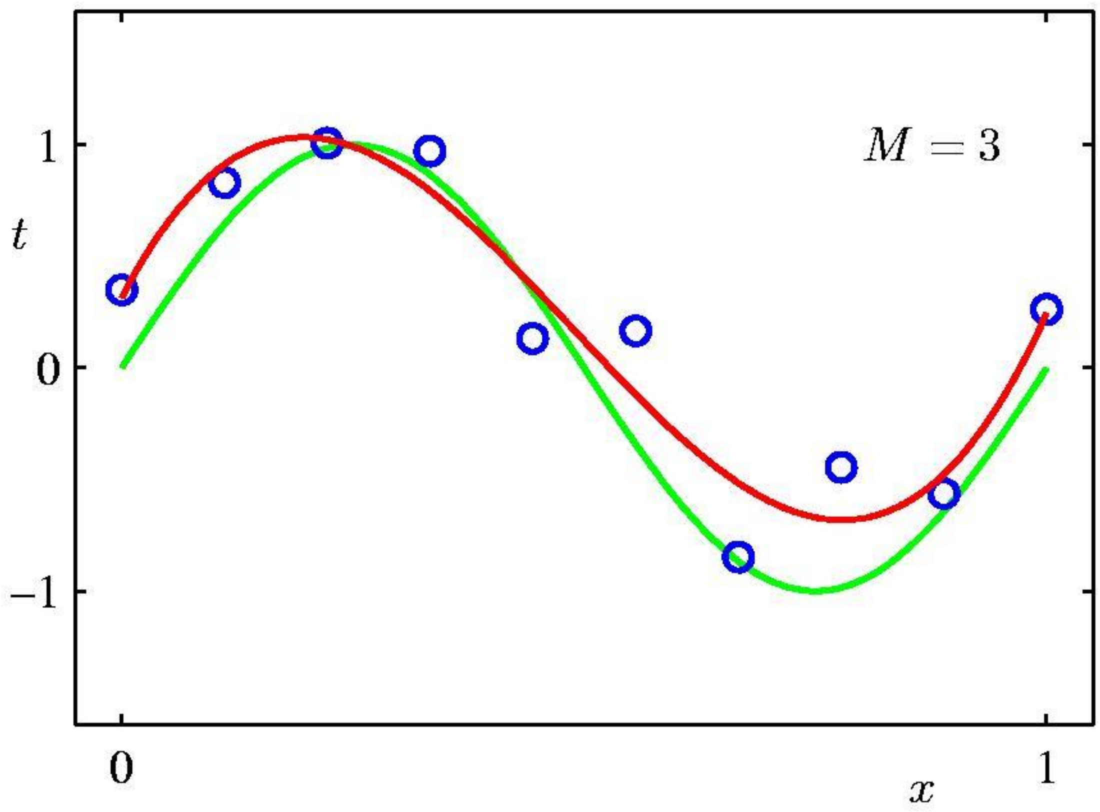
0^{th} Order Polynomial



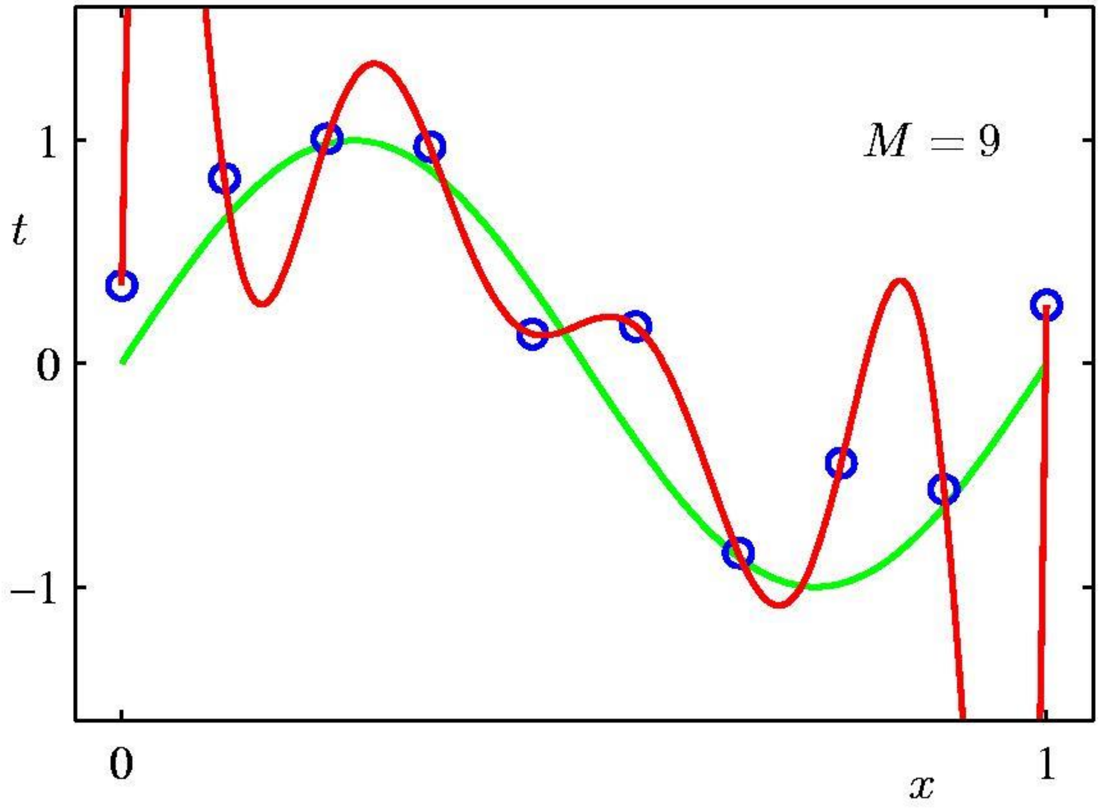
1^{st} Order Polynomial



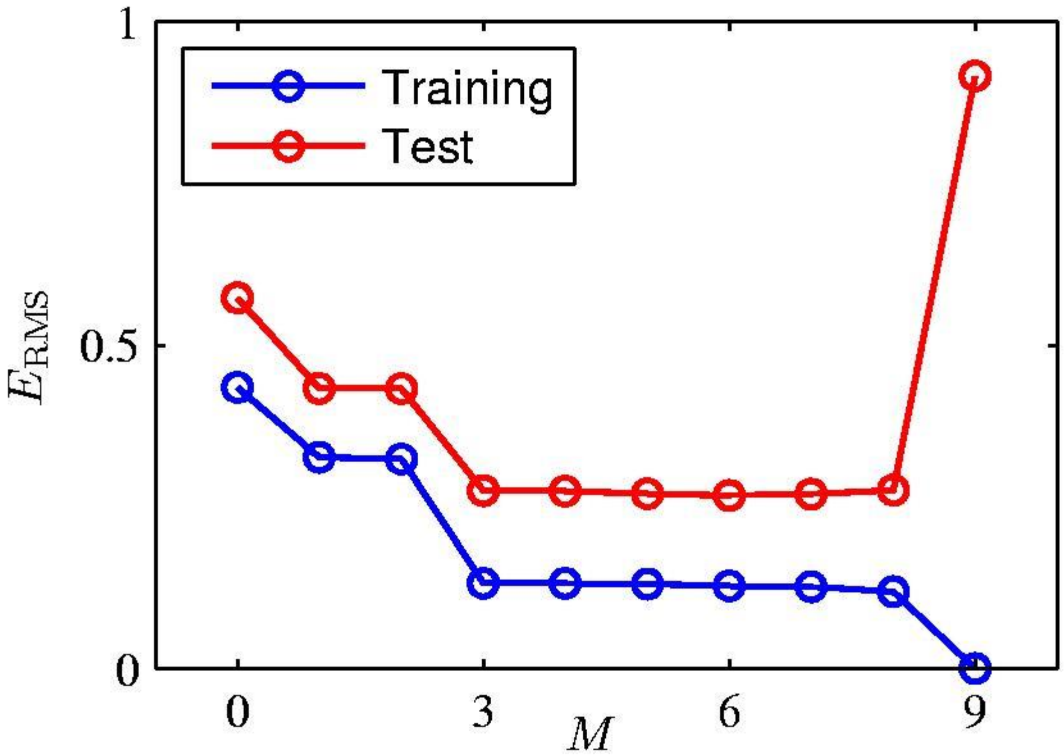
3rd Order Polynomial



9^{th} Order Polynomial



Over-fitting



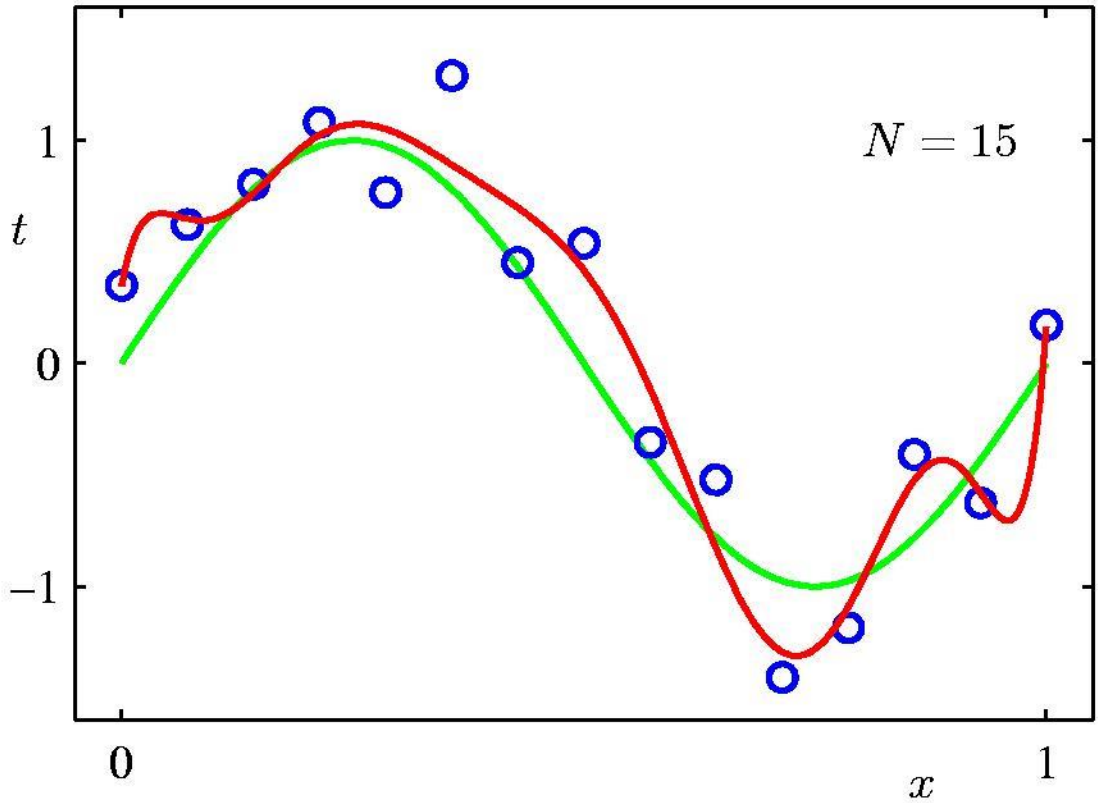
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

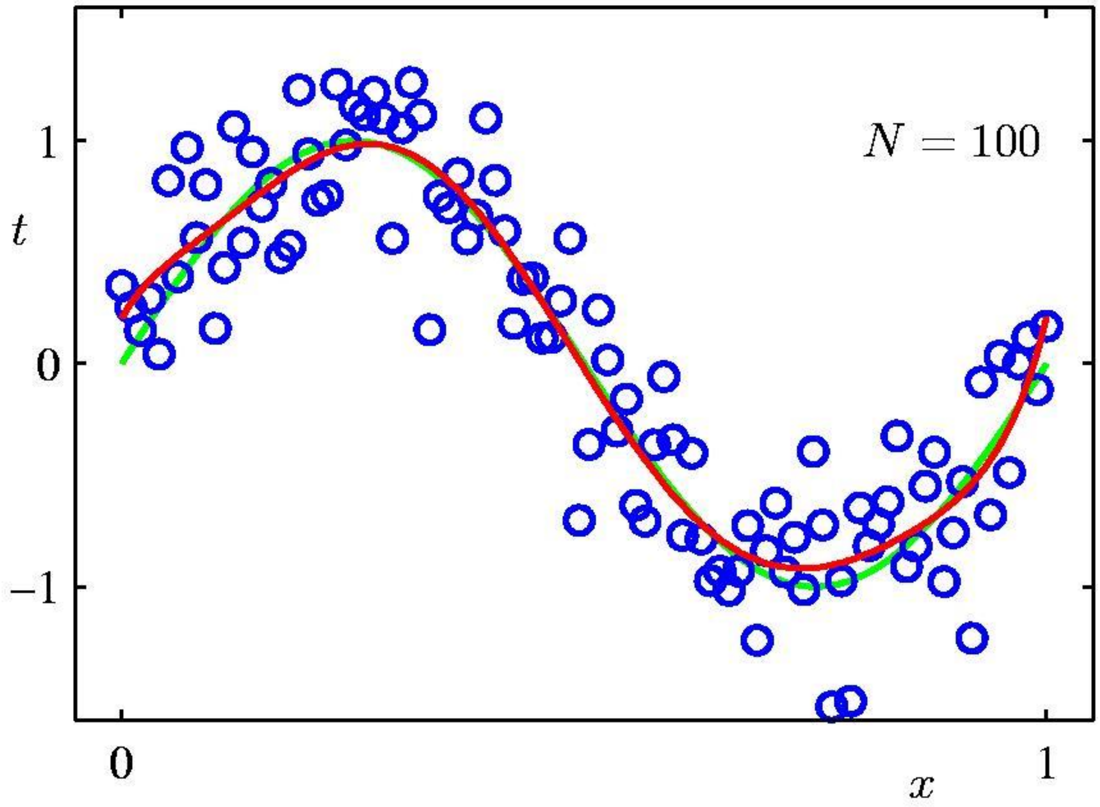
Data Set Size: N=15

9th Order Polynomial



Data Set Size: N=100

9th Order Polynomial



Feature Size vs Sample Size

- Richer feature set is a bless if we have enough data points to make use of these features.
- When the sample size are limited, adding large amounts of feature leads to overfitting.
- An overfitted model performs well for known data points.
- However, its prediction is usually bad for unseen data points.
- Overfitting is a situation that we should always try to avoid.

Regression: Probability-based Perspective

- We derived regression solution based on RSS minimization.
- An alternative, and sometimes more useful perspective, is to view the problem as a probability-based model
- Use maximum likelihood estimator (MLE) to estimate the model.
- For linear regression, RSS minimization and MLE gives the same result.
- But MLE and probability-based perspective are also useful in other extensions.
- Will spend some time on this.

3.1.1 Maximum Likelihood and Least Squares

- Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \qquad \text{where} \qquad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

- which is the same as saying,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

- Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and targets, $\mathbf{t} = [t_1, \ldots, t_N]^{\mathrm{T}}$, we obtain the likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}).$$

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Maximum Likelihood and Least Squares

- Taking the logarithm, we get

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

- where

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

- is the sum-of-squares error.

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}}\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Maximum Likelihood and Least Squares

- Computing the gradient and setting it to zero yields

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

- $\frac{\partial \ln p(\boldsymbol{t}|w,\beta)}{\partial w^T} = 0 = \frac{\partial \beta E_D(w)}{\partial w^T}$
- $= \beta \sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}\phi(x_n)^T$
- $= \beta \sum_{n=1}^{N}\{t_n\phi(x_n)^T - w^T\phi(x_n)\phi(x_n)^T\}$
- Adopting vector notation
- $= \beta\{\boldsymbol{t}^T\Phi - w^T\Phi^T\Phi\} = 0$

$N \times M$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}. = \begin{pmatrix} \phi(\mathbf{x}_1)^T \\ \phi(\mathbf{x}_2)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{pmatrix}$$

Maximum Likelihood and Least Squares

- From $\beta\{\boldsymbol{t}^T\Phi - w^T\Phi^T\Phi\} = 0$
- ➔ $\boldsymbol{t}^{\mathrm{T}}\Phi - w^T\Phi^T\Phi = 0$
- ➔ $\boldsymbol{t}^T\Phi = w^T\Phi^T\Phi$ ➔ $\Phi^T\boldsymbol{t} = \Phi^T\Phi w$
- Solving for w:

$$\mathbf{w}_{\mathrm{ML}} = \overbrace{\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}}^{\text{The Moore-Penrose pseudo-inverse, } \boldsymbol{\Phi}^{\dagger}.}\mathbf{t}$$

- where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}. = \begin{pmatrix} \phi(\mathbf{x}_1)^T \\ \phi(\mathbf{x}_2)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{pmatrix}$$

MLE for Precision (β)

- $\ln p(t|w,\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E_D(w)$
- $E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2$
- $\frac{\partial \ln p(t|w,\beta)}{\partial\beta} = \frac{N}{2}\frac{1}{\beta} - E_D(w) = 0$
- $\frac{N}{\beta} = \sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2$
- $\frac{1}{\beta} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2$
- We can substitute w with $\widehat{w}$, which gives us
- $\frac{1}{\widehat{\beta}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \widehat{w}^T\phi(x_n)\}^2$
- Note that the variance ($1/\hat{\beta}$) computed this way is consistent but biased. An unbiased estimator for variance ($1/\beta$) is $\frac{1}{N-M}\sum_{n=1}^{N}\{t_n - \widehat{w}^T\phi(x_n)\}^2$

3.1.4 Regularized Least Squares (1)

- Consider the error function:

Avoid over-fitting on small training size

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

- With the sum-of-squares error function and a quadratic regularizer, we get

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

- which is minimized by

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

λ is called the regularization coefficient.

Regularized Least Squares (2)

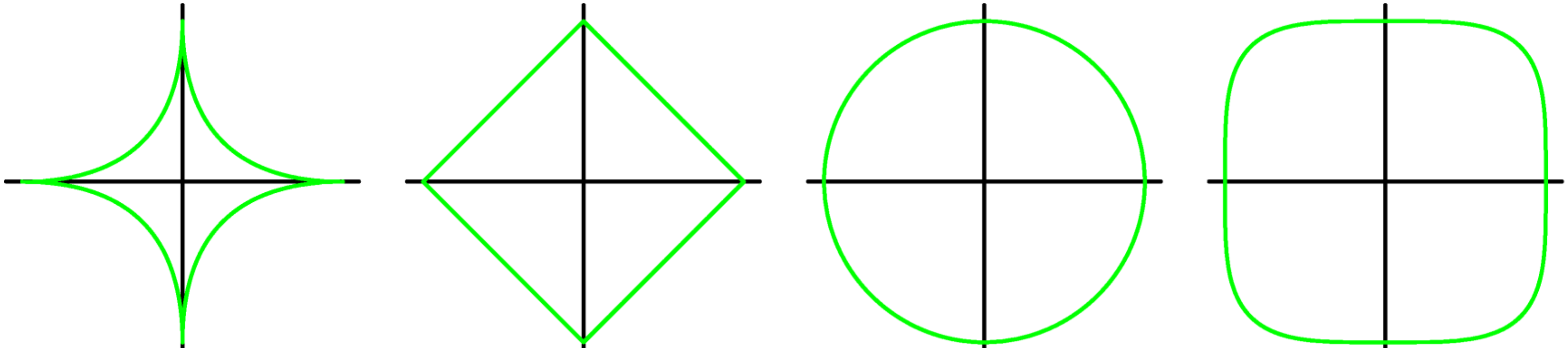
- With a more general regularizer, we have

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

Similar to constraint optimization

$$f(\mathbf{w}) = \tfrac{1}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\right\}^2$$

$$\text{subject to } \sum_{j=1}^{M}\left|w_j\right|^q \le \eta$$



$q = 0.5$ $q = 1$ $q = 2$ $q = 4$

Lasso

Quadratic (Ridge Regression)

Ridge Regression Example

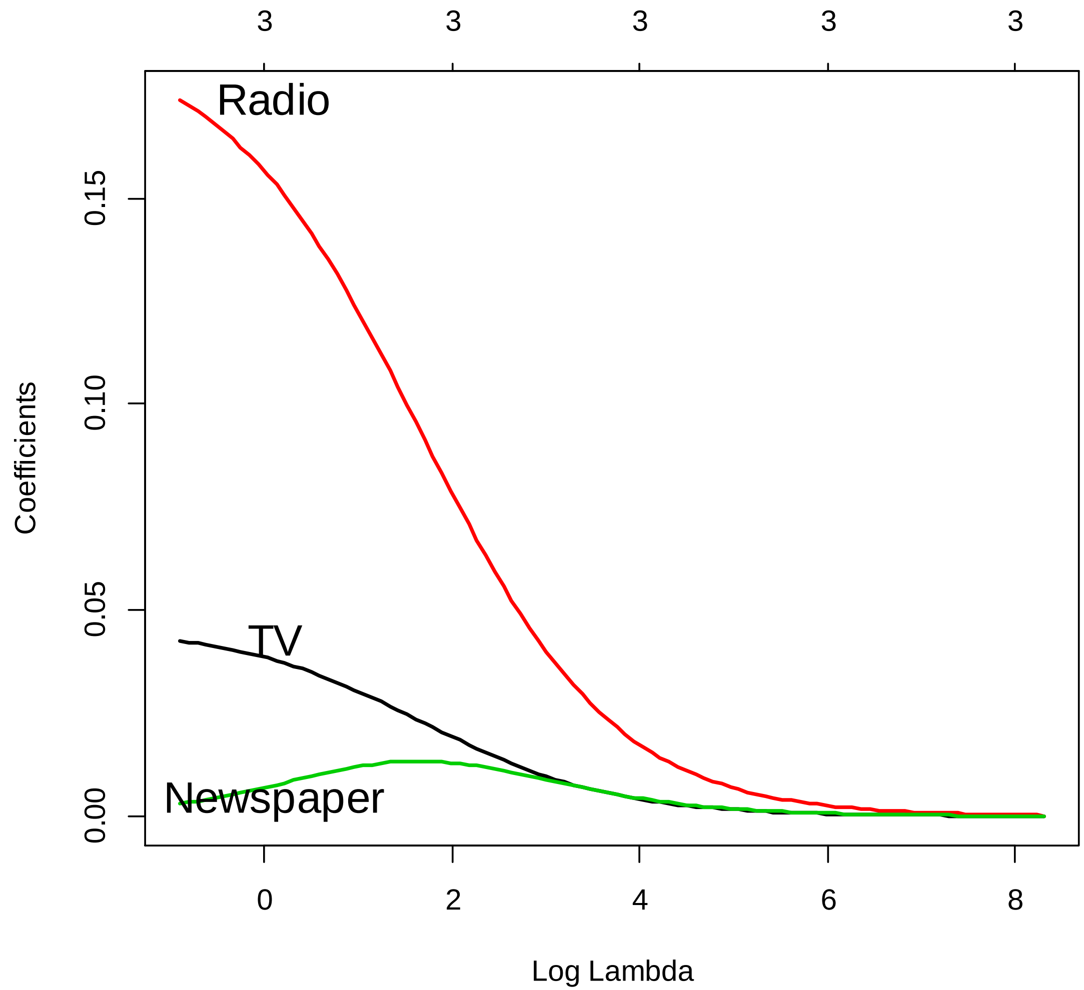
- Fitting data via minimizing $\frac{1}{2}\sum\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \frac{\lambda}{2}\sum w_j^2$
- Changing λ can gives different fitted $\boldsymbol{w}$
- Using glmnet package to demonstrate the relationship between λ and $\boldsymbol{w}$.

```
import glmnet_python
from glmnet import glmnet
from glmnetPlot import glmnetPlot

df1 = pd.read_csv('Advertising.csv')
allfeatures = ['TV', 'Newspaper', 'Radio']
fit = glmnet(x = df1[allfeatures].as_matrix(),
             y = df1['Sales'].as_matrix(),
             alpha = 0, family = 'gaussian')
glmnetPlot(fit, xvar = 'lambda', label = True)
```

Ridge Regression Example

- As lambda increases, all coefficients become smaller
- When lambda is large enough, all coefficients are essentially 0.
- The relationship between lambda and a coefficient is not always monotonous.
- We need to find a best lambda when constructing a prediction model.

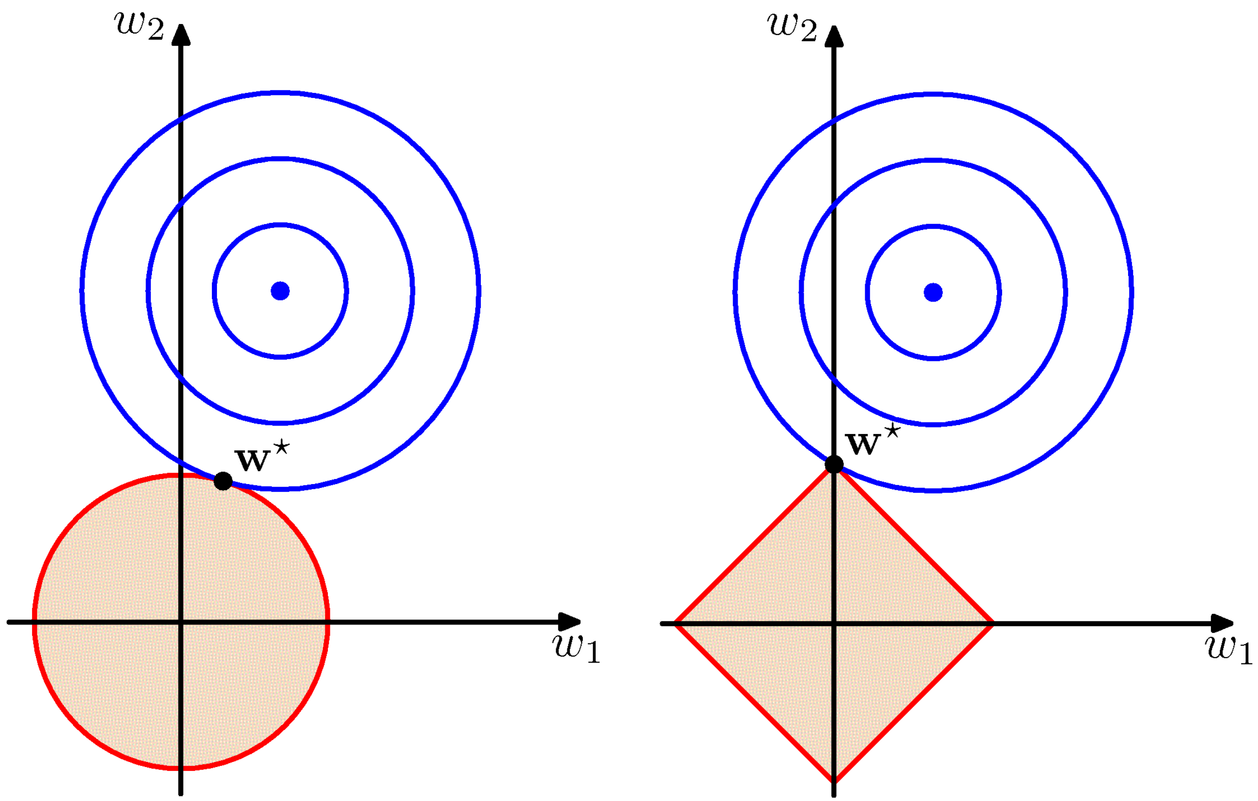


Note: glmnet_py currently only works for Linux (e.g. Ubuntu)

Regularized Least Squares (3)

- Lasso tends to generate sparser solutions than a quadratic regularizer.

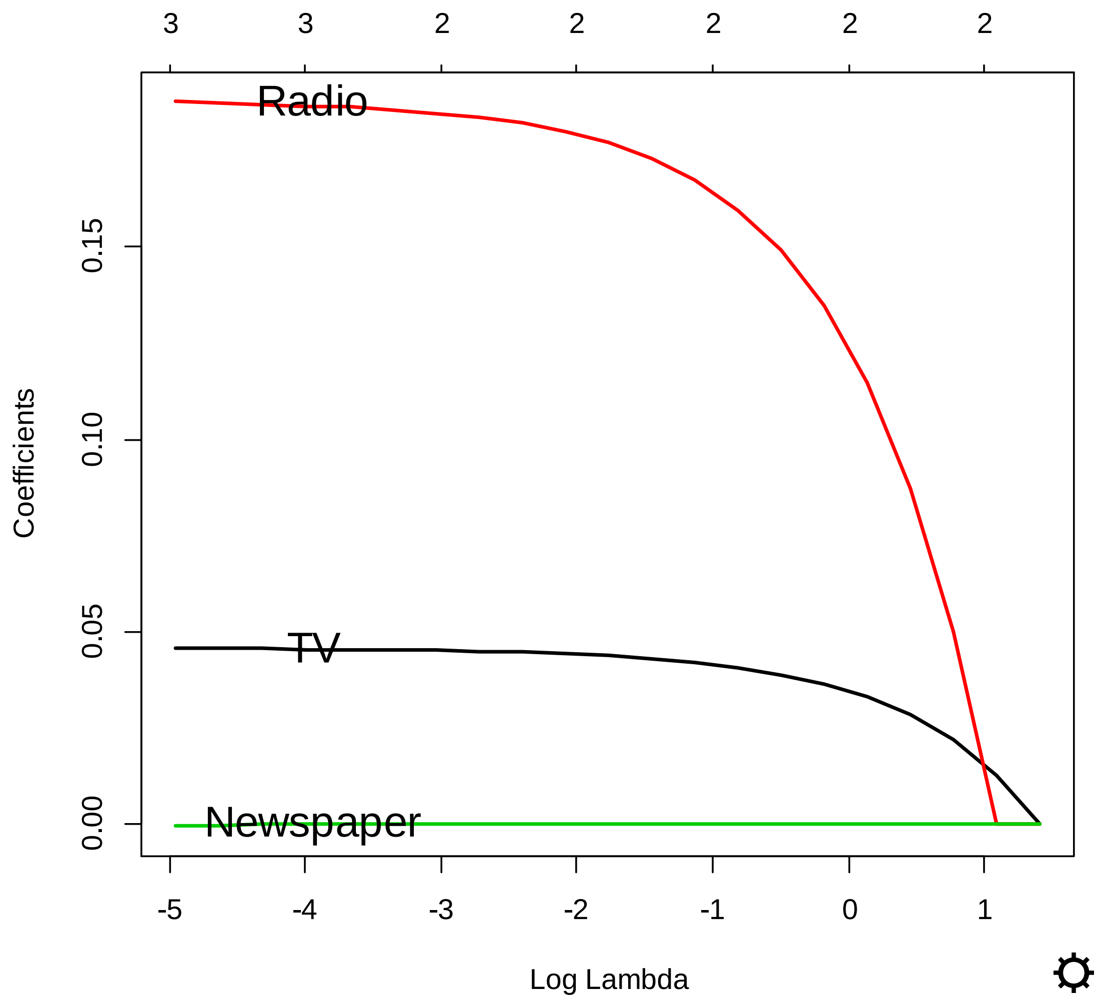
If λ is sufficiently large



Lasso Regression Example

Lasso regression loss: $\frac{1}{2}\sum\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \lambda\sum\left|w_j\right|$

- Newspaper was set to zero for all lambda.
- Radio becomes zero when lambda is large enough
- Can be used to select # of parameters included.



Model Training: Lasso Regression

- Lasso regression loss:

$$\mathrm{L} = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \lambda\sum_{i=1}^{M}|w_i|$$

- We are trying to minimize L by choose the best $\boldsymbol{w}$.
- However, this problem has no closed-form solution.
- The estimation is based on the coordinate descent algorithm.
- That is, we loop through each w_i and solve for a better value that can improve upon current value.

Coordinate Descent and Soft Thresholding

$$\mathrm{L} = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \lambda\sum_{i=1}^{M}|w_i|$$

- Assume that we start from $\widetilde{\boldsymbol{w}}$ for $\boldsymbol{w}$.
- For each w_j, j=1, 2, …, M, we try to find a better value that can decrease L (fixing other w_i).
- We solve the problem by looking for a w_j that can give us zero gradient w.r.t. w_j.
- $\frac{\partial L}{\partial w_j} = \sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)\left(-\phi(\boldsymbol{x}_n)_j\right) + \lambda sgn(w_j)$.
- Here $sgn(w_j) = 1$ if $w_j > 0$,
  $sgn(w_j) = -1$ if $w_j < 0$
  $sgn(w_j) = 0$ if $w_j = 0$

Coordinate Descent and Soft Thresholding (Cont'd.)

- To facilitate subsequent discussion, we write $\boldsymbol{w}^T\phi(\boldsymbol{x}_n)$ as
- $\boldsymbol{w}^T\phi(\boldsymbol{x}_n) = w_1\phi(\boldsymbol{x}_n)_1 + w_2\phi(\boldsymbol{x}_n)_2 + \cdots + w_M\phi(\boldsymbol{x}_n)_M = w_1\phi_{n,1} + w_2\phi_{n,2} + \cdots w_M\phi_{n,M} = \boldsymbol{w}_{-j}^T\boldsymbol{\phi}_{n,-j} + w_j\phi_{n,j}$.
- Here $\boldsymbol{w}_{-j}^T = [w_1\ w_2\ \ldots\ w_{j-1}\ w_{j+1}\ \ldots w_M]$
- Similarly, we define $\boldsymbol{\phi}_{n,-j}$ as the vector that exclude the j-th element from $\boldsymbol{\phi}_n$

Coordinate Descent and Soft Thresholding (Cont'd.)

- Using the new notation,
- $\frac{\partial L}{\partial w_j} = \sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)\left(-\phi(\boldsymbol{x}_n)_j\right) + \lambda sgn\left(w_j\right)$
- $= \sum_{n=1}^{N}\left(t_n - \boldsymbol{w}_{-j}^T\boldsymbol{\phi}_{n,-j} - w_j\phi_{n,j}\right)\left(-\phi_{n,j}\right) + \lambda sgn\left(w_j\right)$
- $= \sum_{n=1}^{N}\left(t_n - \boldsymbol{w}_{-j}^T\boldsymbol{\phi}_{n,-j}\right)\left(-\phi_{n,j}\right) + w_j \sum_{n=1}^{N} \phi_{n,j}^2 + \lambda sgn\left(w_j\right)$
- Since we are using $\widetilde{\boldsymbol{w}}$ as the starting value, we are solving
- $\sum_{n=1}^{N}\left(t_n - \widetilde{\boldsymbol{w}}_{-j}^T\boldsymbol{\phi}_{n,-j}\right)\left(-\phi_{n,j}\right) + w_j \sum_{n=1}^{N} \phi_{n,j}^2 + \lambda sgn\left(w_j\right) = 0$
- $\Rightarrow w_j = \frac{\sum_{n=1}^{N}\left(t_n - \widetilde{\boldsymbol{w}}_{-j}^T\boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} - \lambda sgn\left(w_j\right)}{\sum_{n=1}^{N} \phi_{n,j}^2}$

Coordinate Descent and Soft Thresholding (Cont'd.)

- That is, give $\tilde{\boldsymbol{w}}$, we want to update w_j by
- ➔ $w_j = \dfrac{\sum_{n=1}^{N}\left(t_n - \tilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} - \lambda sgn(w_j)}{\sum_{n=1}^{N}\phi_{n,j}^2}$
- However, w_j is on both sides…
- If the resulting w_j is positive,
- that is if $\dfrac{\sum_{n=1}^{N}\left(t_n - \tilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} - \lambda}{\sum_{n=1}^{N}\phi_{n,j}^2} > 0,$
- then $w_j = \dfrac{\sum_{n=1}^{N}\left(t_n - \tilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} - \lambda}{\sum_{n=1}^{N}\phi_{n,j}^2}$
- Do similar things for negative w_j.

Coordinate Descent and Soft Thresholding (Cont'd.)

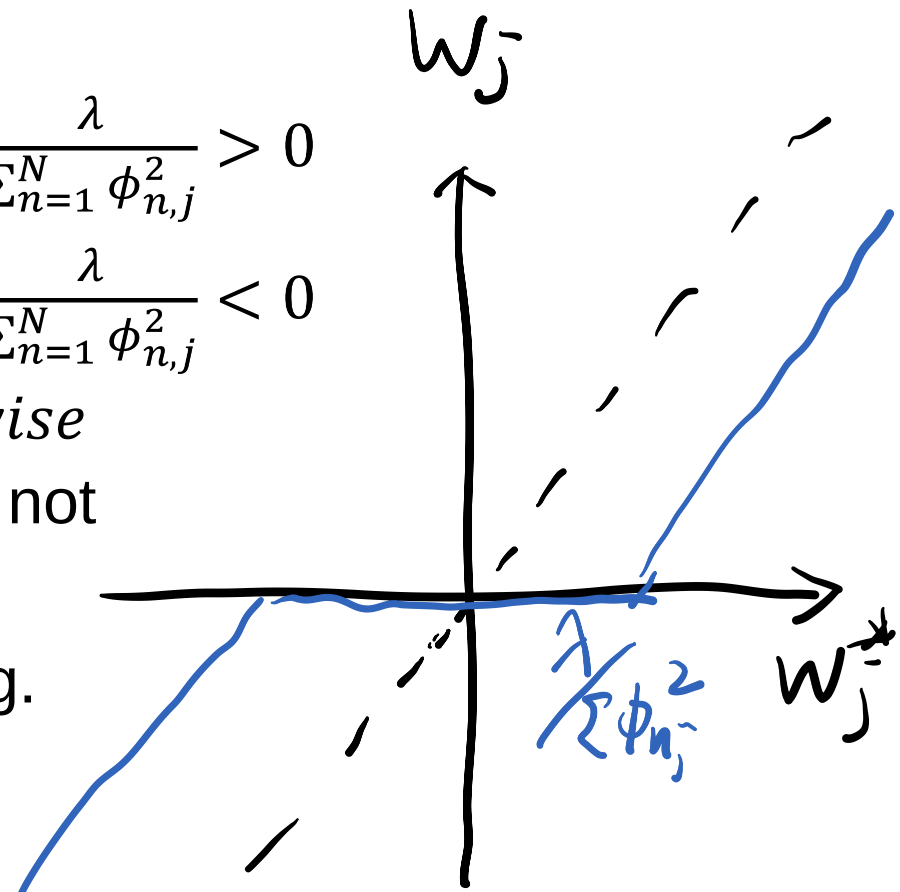
- To summarize:
- if $\frac{\sum_{n=1}^{N}\left(t_n - \widetilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} - \lambda}{\sum_{n=1}^{N}\phi_{n,j}^2} > 0,$
- then $w_j = \frac{\sum_{n=1}^{N}\left(t_n - \widetilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} - \lambda}{\sum_{n=1}^{N}\phi_{n,j}^2}$
- if $\frac{\sum_{n=1}^{N}\left(t_n - \widetilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} + \lambda}{\sum_{n=1}^{N}\phi_{n,j}^2} < 0,$
- then $w_j = \frac{\sum_{n=1}^{N}\left(t_n - \widetilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j} + \lambda}{\sum_{n=1}^{N}\phi_{n,j}^2}$
- Otherwise, $w_j = 0$

Coordinate Descent and Soft Thresholding (Cont'd.)

- If we set $w_j^* = \frac{\sum_{n=1}^{N}\left(t_n - \tilde{\boldsymbol{w}}_{-j}^T \boldsymbol{\phi}_{n,-j}\right)\phi_{n,j}}{\sum_{n=1}^{N} \phi_{n,j}^2}$, then

- $$w_j = \begin{cases} w_j^* - \frac{\lambda_j}{\sum_{n=1}^{N} \phi_{n,j}^2} & if\ w_j^* - \frac{\lambda}{\sum_{n=1}^{N} \phi_{n,j}^2} > 0 \\ w_j^* + \frac{\lambda_j}{\sum_{n=1}^{N} \phi_{n,j}^2} & if\ w_j^* + \frac{\lambda}{\sum_{n=1}^{N} \phi_{n,j}^2} < 0 \\ 0 & otherwise \end{cases}$$

- In words: set w_j to 0 if w_j^* it is not too far from zero.
- This is called soft thresholding.



Lasso Training

- Start from an initial $\tilde{\boldsymbol{w}}$, iteratively update each w_j using the soft thresholding equation until converge.

Hyper-parameter Tuning

- There are three variation of the models
- Ridge regression: $\frac{1}{2}\sum\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \frac{\lambda}{2}\sum w_j^2$
- Lasso regression: $\frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \lambda\sum_{i=1}^{M}|w_i|$
- Elastic-net regression (fixed α) :
  $\frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 + \lambda\left(\alpha\sum_{i=1}^{M}|w_i| + \frac{(1-\alpha)}{2}\sum w_j^2\right)$.
- For each model, need to tune λ for the best prediction performance. You have several options:
- 1. Brute force method: grid search λ
- 2. Use the glmnet algorithms
- 3. Use theoretical value implied by the evidence function (will not cover, for ridge regression only)

Discussion

- We know that ridge, lasso, and elastic-net performs better than OLS in most situations. Why? What is the intuition behind the better performance?